Atty. Docket No. MS305314.1/MSFTP596US

TRAINING FILTERS FOR IP ADDRESS AND URL LEARNING

by

Joshua T. Goodman, Robert L. Rounthwaite, Geoffrey J. Hulten, and Wen-tau Yih

MAIL CERTIFICATION

I hereby certify that the attached patent application (along with any other paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on this date March 25, 2004, in an envelope as "Express Mail Post Office to Addressee" Mailing Label Number EV373131597US addressed to the Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Himanshu S. Amin

Title: TRAINING FILTERS FOR IP ADDRESS AND URL LEARNING

CROSS REFERENCE TO RELATED APPLICATION

This application is a continuation-in part of U.S. Application Serial No. 10/454,168, entitled *Origination/Destination Features and Lists For Spam Prevention* and filed on June 4, 2003, the entirety of which is incorporated herein by reference.

TECHNICAL FIELD

This invention is related to systems and methods for identifying both legitimate (*e.g.,* good mail) and undesired information (*e.g.*, junk mail), and more particularly to detecting spam messages in part by scanning messages using a filter trained on IP address or URL features and another filter independently trained on text-related features and/or other features extractable from a message.

BACKGROUND OF THE INVENTION

The advent of global communications networks such as the Internet has presented commercial opportunities for reaching vast numbers of potential customers. Electronic messaging, and particularly electronic mail ("email"), is becoming increasingly pervasive as a means for disseminating unwanted advertisements and promotions (also denoted as "spam") to network users.

The Radicati Group, Inc., a consulting and market research firm, estimates that as of August 2002, two billion junk e-mail messages are sent each day - this number is expected to triple every two years. Individuals and entities (*e.g.*, businesses, government agencies) are becoming increasingly inconvenienced and oftentimes offended by junk messages. As such, spam is now or soon will become a major threat to trustworthy computing.

A key technique utilized to thwart spam is employment of filtering systems/methodologies. One proven filtering technique is based upon a machine learning approach - machine learning filters assign to an incoming message a probability that the message is spam. In this approach, features typically are extracted from two classes of

example messages (*e.g.*, spam and non-spam messages), and a learning filter is applied to discriminate probabilistically between the two classes. Since many message features are related to content (*e.g.*, words and phrases in the subject and/or body of the message), such types of filters are commonly referred to as "content-based filters".

With the onslaught of such spam filtering techniques, many spammers have thought of ways to disguise their identities to avoid and/or bypass spam filters. Thus, conventional content-based and adaptive filters may become ineffective in recognizing and blocking disguised spam messages.

SUMMARY OF THE INVENTION

The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is not intended to identify key/critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

The present invention provides for a system and method that facilitate distinguishing between spam and good messages in part by employing various smoothing techniques to place less emphasis on making weights associated with origination or destination features small and more emphasis on making weights associated with words small. As a result, a machine learning filter can rely more on IP addresses, for example, since they may need to see them fewer times than words to give them a higher weight.

According to one aspect of the present invention, the system and method provide for a machine learning filter that utilizes origination features such as IP addresses and/or destination features such as URLs apart or independently from other features such as text-based features (*e.g.*, key words or phrases). In particular, at least two filters can be trained independently of the other to handle different features. For example, a first filter can be trained on only origination or destination features and any other relevant features can be learned by a second filter, such that the training of the first filter does not influence the training of the second filter, or vice versa.

Messages can be subjected to at least one of the two filters, depending on the particular feature in focus or the most relevant feature in the particular message. For instance, imagine that the feature in focus is an IP address or a URL. Thus, one of the at least two filters can be trained on IP addresses or URLs and the other on non-IP address or non-URL features (*e.g.,* text, key words, character strings, etc.).

Spammers typically send out the same message over and over again from a known IP address; however every few weeks, spammers can send the same message but from a different or new IP address. Had the IP address stayed the same on the current message, one filter trained only on known IP addresses (*e.g.,* known to be spam or known to be good) could have readily identified the message as spam. However, because the IP address is now unknown, a filter trained only on text-related or other features can be employed to determine whether the message is spam. Thus, two different filters can be trained for different features. As a result, the present invention can better generalize to cases when the IP address or URL is known or unknown.

Alternatively or in addition, the two filters can be used together either concurrently or consecutively to scan for the different features without any undue influence from or reliance on the other. For example, imagine that a spammer is sending "get out of debt" spam from a particular IP address and imagine that a filter has learned that this IP address is a near perfect indicator of whether such messages are spam. In addition, a second filter has learned that key words or phrases such as "out of debt" "debt" "relief" and "free quote" included in the message are also strongly indicative of spam. By combining the results of both filters, a spam prevention system can better distinguish between legitimate messages and spam, thereby improving spam catch rates.

Another approach involves training multiple models of a filter. For example, one can be trained using IP addresses, another can be trained using the top (first) 24 bits of IP addresses, and yet another can be trained using the top 16 bits of IP addresses (or some number of bits less than 32 bits). For instance, if there is insufficient data on a particular IP address but the filter or model has observed several messages with the top 24 bits of that IP address, the model trained using the top 24 bits can be selected to scan or "filter" the IP address data that is on hand. Thus, multiple models can be trained such that only a

specific model for which there is sufficient training data is used. This is also applicable to domain names.

According to a further aspect of the invention, the ambiguity of origination or destination features in messages can be resolved to a large extent so that a machine learning filter can appropriately and accurately identify the relevant features to determine whether the message is spam. This can referred to as the ambiguity problem.

The ambiguity problem can arise with respect to URLs as well as IP addresses. Any number of URLs can be added to a spam message in order to mislead and confuse a filter. A conventional filter may not be able to tell which URL is the relevant destination feature of the message. Hence, an accurate determination as to whether the message is spam can be compromised and perhaps unattainable the majority of the time. With respect to IP addresses, most clients cannot decipher which is the last IP address external to the system, the identification of which can be important to classify the message as spam or legitimate. To address either situation, various heuristics and/or algorithmic functions can be employed to learn the worst scoring IP address as well as to optimize or maximize accuracy of training data. In addition, user feedback as to which messages are marked as spam and which are marked as good can also be collected and used as input for a machine learning filter to at least facilitate determining which IP addresses or URLs appear to be spam or legitimate.

As previously mentioned, the IP address and/or URL features may be the most valuable features extractable from a message for spam classification and prevention. However, machine learning filters can have no way of knowing this and in most cases, may assign equal or comparable weights to an IP address in a message as well as to the words in the message. Most machine learning systems provide some type of technique to smooth their parameters. The present invention also facilitates modifying the smoothing or regularization techniques so that IP addresses and URLs are counted as more valuable.

Furthermore, it should be appreciated that various aspects of the present invention can be used in conjunction with a feedback loop, in which users who opt-in provide feedback regarding messages they have received to a machine learning system to affect filter updates and the like.

To the accomplishment of the foregoing and related ends, certain illustrative aspects of the invention are described herein in connection with the following description and the annexed drawings. These aspects are indicative, however, of but a few of the various ways in which the principles of the invention may be employed and the present invention is intended to include all such aspects and their equivalents. Other advantages and novel features of the invention may become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a high-level block diagram of a spam prevention system that can select at least one of at least two filters which are independently trained using different features to detect spam in accordance with an aspect of the present invention.

Fig. 2 is a block diagram of a filtering system that provides employment of at least a combination of two different filters that can be utilized together to detect spam in accordance with an aspect of the present invention.

Fig. 3 is a schematic flow diagram of an exemplary method for training at least two separate machine learning filters that are independent of the other in accordance with an aspect of the present invention.

Fig. 4 is a schematic flow diagram of an exemplary method for determining which feature trained-filter or which model of filter to employ determines whether a message is spam in accordance with an aspect of the present invention.

Fig. 5 is a flow diagram of an exemplary method for distinguishing between internal and external IP addresses to facilitate determining the last external IP address in accordance with an aspect of the present invention. The method is also applicable to URLs.

Fig. 6 is a flow diagram of an exemplary method for distinguishing between internal and external IP addresses to facilitate determining the last external IP address in accordance with an aspect of the present invention. The method is also applicable to URLs.

Fig. 7 is a flow diagram of an exemplary method that facilitates performing machine learning for functions that are not smoothly differentiable with respect to

messages and IP addresses and features of IP addresses in accordance with an aspect of the present invention.

Fig. 8 is a schematic diagram of machine learning systems and their smoothing can be altered so that IP addresses and URLs are given more value and counted as more valuable.

Fig. 9 is a schematic block diagram of an exemplary communication environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It may be evident, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the present invention.

As used in this application, the terms "component" and "system" are intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers.

The subject invention can incorporate various inference schemes and/or techniques in connection with generating training data for machine learned spam filtering. As used herein, the term "inference" refers generally to the process of reasoning about or inferring states of the system, environment, and/or user from a set of observations as captured *via* events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic – that is, the computation of a probability

distribution over states of interest based on a consideration of data and events. Inference can also refer to techniques employed for composing higher-level events from a set of events and/or data. Such inference results in the construction of new events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the events and data come from one or several event and data sources.

It is to be appreciated that although the term message is employed extensively throughout the specification, such term is not limited to electronic mail *per se*, but can be suitably adapted to include electronic messaging of any form that can be distributed over any suitable communication architecture. For example, conferencing applications that facilitate a conference or conversation between two or more people (*e.g.*, interactive chat programs, and instant messaging programs) can also utilize the filtering benefits disclosed herein, since unwanted text can be electronically interspersed into normal chat messages as users exchange messages and/or inserted as a lead-off message, a closing message, or all of the above. In this particular application, a filter can be trained to automatically filter particular message content (text and images) in order to capture and tag as junk the undesirable content (*e.g.*, commercials, promotions, or advertisements).

In the subject invention, the term "recipient" refers to an addressee of an incoming message. The term "user" refers to a recipient who makes use of messaging programs as well as filtering systems to detect and prevent spam and/or who has chosen, either passively or actively, or has been indirectly selected to participate in the feedback loop systems and processes as described herein.

One of the primary tasks of filtering systems can be to improve catch rates and to increase gains in spam detection. According to an aspect of the present invention, changing the smoothing of particular message features can facilitate increasing gains in spam detection. Smoothing can refer to an amount of trust for a particular feature. In general, smoothing or regularization techniques try to keep weights of features small. Since the weights of features of a message, for example, are added together to compute a score, a small weight means it contributes very little to the final score.

For instance, a random odd word in a message may just be there by chance. Therefore, it does not mean that the word can be trusted to indicate or identify either

good or spam messages just because it has been seen in a few spam messages or a few good messages. However, the IP address indicates where the message comes from and can be difficult to forge. Furthermore, most IP addresses are pure: pure good or pure spam. Thus, if the same IP address is seen a few times, one can be pretty certain that it can identify either good or bad messages, as the case may be. By changing the smoothing such that less emphasis is put on making the weight of an IP address small or that more emphasis is put on making the weight of words small, the filter can rely more on the IP address. Because machine learning systems can see IP addresses fewer times than words to understand that it indicates good messages or spam, the IP addresses can be given a higher weight and thus more value can be added to the overall determination of whether a message is spam. As shown by experimentation of this, substantial increases in gains were observed.

Moreover, origination features like IP addresses and destination features like URLs are both very powerful parts of a message because they cannot be easily faked by spammers. As a result, they can be very powerful features for a spam filter and in particular, a machine learning based spam filter. Unfortunately, a generalization problem can exist. In particular, the IP address and/or URL features can change from day to day; thus filters trained to emphasize IP addresses over words can be ineffective and useless.

For instance, a spammer who sells mortgage refinancing can change his IP address from week to week. Some messages about mortgage refinancing are legitimate, although the vast majority is spam. If a filter has been trained to learn key words such as "mortgage" "refinance" "no" "obligation" and "free" as well as IP addresses, then it can determine to ignore all of those key words common to the mortgage-type messages and only look at the IP addresses because that may be the only feature different among the messages. This means that the filter may not factor the presence of the above key words in to its decision as to whether the message is most likely to be spam. When the spammer changes his IP address and sends the mortgage message again, the filter may not generalize to the new message because it does not recognize the new IP address. In the meantime, it can take a few weeks to update the filter and this delay can lead to the delivery of undesirable amounts of spam.

Figs. 1-4 which follow below address the generalization to new messages in accordance with an aspect of the present invention. In Fig. 1, there is illustrated a high-level block diagram of a system 100 that facilitates spam prevention in part by training at least two different filters independently of each other using different feature types for each filter. The system 100 includes a feature extraction component 110 that can extract one or more features including origination or destination features as well as text or word related features from an incoming message(s).

The extracted data can be analyzed or organized by a feature analysis component 120 into two or more sets or subsets of training data for use with a respective training component 130. For instance, the extracted data can include IP addresses, the top 24 bits of IP addresses, the top 16 bits of IP addresses, URLs as well as domain names (fully qualified domain names and the sub-domains thereof), header line information, words, phrases, and/or character strings. Thus, the set of training data can comprise a first subset comprising IP address data, a second subset comprising the top 24 bits of IP addresses, a third subset comprising the top 16 bits of IP addresses, a fourth subset comprising URL data, and/or a fifth subset comprising substantially all other relevant text-related data.

The filter training component 130 can include a plurality of subcomponents that can individually train a first filter 140 on a particular subset of training data which is separate and apart from the training of a second filter 150 using a different subset of training data. For example, the first filter 140 can be trained on the IP address data (160) (or URL data 170). Meanwhile, the second filter 150 can be trained on the text-related features. As a result, the two filters are trained independently of the other such that the training of the one filter does not unduly influence the training of the other. Thus, when spammers change their IP addresses every few weeks, the IP address is not blindly relied upon to detect for spam. Instead the text-feature trained filter (150) can be employed to catch it.

Moreover, multiple filters can be trained, whereby each filter is trained on a different feature type to facilitate generalizing to new variations of features in spam messages. For example, when a full 32 bits of an IP address are not discernible, filters can be trained on less than 32 bits of the IP addresses. One filter can be trained on the top 24 bits of IP addresses, another filter can be trained on the top 16 bits of IP addresses,

still another filter can be trained on URL data, etc. The same can be done with respect to domain names (sub-domains of fully qualified domain names). As a result, the most specific filter or model can be used based on the most amount of information available in an incoming message. For instance, if there is insufficient data on a particular IP address extracted from the message, but the top 16 bits of that IP address has been observed in previous messages, the filter trained on the top 16 bits can be employed since this is the most specific filter available based on the available data.

It should be appreciated that such specific filters can include or exclude other features and in particular, text-based features. For example, the filter trained on the top 16 bits of an IP address can also be trained on text-based features such as keywords, phrases, and the like. Conversely, the 16-bit filter can forgo training on text-based features. In such a case, an additional filter trained on the text-based features may be employed to filter the incoming messages.

In an alternative scheme, two or more different machine learning filters can be trained and employed concurrently or consecutively to more accurately detect spam – for example, one using at least one of the IP address features and the URLs, and the other using other relevant features. If these are probabilistic models, this is equivalent to assuming independence between the features in one model and the features in the other model. This can be somewhat unintuitive – one would expect the source of spam to be highly correlated with its content, so an explicit assumption of independence is surprising. Some model types (Naïve Bayes) assume independence between all features, and adding IP addresses and URLs as independent features to such a model is natural. Naïve Bayes models typically assume independence not because such independence is true, nor because it improves the model, but primarily because when *all* features are assumed independent, modeling becomes very simple. However, Naïve Bayes models typically have mediocre performance for spam filtering when compared to other model types (*e.g.,* support vector machines (SVMs), perceptrons, maximum entropy (a.k.a. logistic regression), neural networks) that explicitly model dependence.

One of these more powerful model types can be used for modeling non-IP, non-domain features. When the base model type used for other features is powerful enough to model dependencies, it is very surprising to explicitly model either the URLs or the IP

addresses as independent of the other features. The two or more models so trained can then be combined in multiple ways. If both models return scores of some sort (*e.g.*, probabilities), the scores can be added together or multiplied together. Alternatively, an additional model such as a weighted linear combination, for example, can be trained to combine the scores.

Referring now to Fig. 2, there is illustrated a schematic diagram of an exemplary path 200 that a message 205 can take upon its arrival according to an aspect of the present invention. In particular, Fig. 2 demonstrates the utilization of two independently trained filters 210 and 220 when detecting for spam. Upon its arrival, the message 205 can be subjected to an examination by one or more feature-specific filters to determine whether the message is spam or legitimate. For instance, in the prior example where the spammer changed his IP address, the next time the message is received, it can be from a different or unknown IP address. To combat such an occurrence, two filters can be trained: one for the case where the IP address is known 230 and one for the case where the IP address is unknown 240. In addition, a filter trained only on other relevant features (*e.g.*, text-based features) can also be trained (220).

In this instance, the text trained filter 220 can be applied to the message together with the unknown IP address filter 240 in order to determine whether the message is spam since the IP address extracted from the message is unknown. The unknown IP address filter can be trained by using other messages having unknown IP addresses which were received during a particular time frame. For example, messages also having unknown IP addresses that were received at about the same time as the current message can be used.

Unknown IP addresses can be viewed at several levels. For example, an unknown IP address can be one that has never been seen in its exact form; however, the top 16 or 24 bits of the IP address may have been observed before. Thus, a filter trained only on the top 24 bits can be employed. If only the top 16 bits of the IP address in the message have been seen before, then the 16-bit filter can be used – with or without the text-related filter. However, if the top 16 bits have never been seen before, then the text-related filter can be used alone. Thus, the filtering system can be optimized to generalize to employ the most specific filter or model for which there is sufficient data.

Various methodologies in accordance with the subject invention will now be described via a series of acts. It is to be understood and appreciated that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in different orders and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the present invention.

Referring now to Fig. 3, there is illustrated a flow diagram of an exemplary method 300 that facilitates building feature-specific filters separately from one another for employment in a spam prevention system. In particular, the method 300 involves obtaining training data at 310. The training data can comprise any number of messages that may be known to be either spam or good messages. At 320, one or more origination or destination features can be extracted from the message(s). Such features include IP address data (origination) and URL data (destination). At 330, a first filter can be trained using only the IP address or URL data. Meanwhile at 340, other relevant features such as text-related features can be extracted from the message(s) and then used to train a second filter at 350. It should be appreciated that the text-related features can also be extracted at about the same time as the origination and destination features at 320.

As can be seen in the figure, the second filter trained only on non-origination and non-destination features is trained separately and independently from the origination or destination-based filter. This mitigates undue influence from either feature over the other when examining messages for spam-like qualities and characteristics, which can be particularly pertinent when spam messages have changed slightly from previous known versions.

Though not depicted in the figure, extracting and filtering URL data presents additional challenges since a message can have multiple URLs included therein. Some URLs may be known and some may be unknown. In view of this, one filter for the known URLs can be employed and a different filter can be employed when at least one URL is unknown in the message.

Moreover, the present invention assumes independence between an IP address (or URL) and other features such as text features. However, the invention can assume dependence between text features. For example, the system or method as described in Figs. 1-3, *supra*, can know that "click" and "here" go together. That is, independence is not assumed between "click" and "here" and thus, these dependencies can be modeled accordingly. On the contrary, independence is assumed between the IP address the "click here" comes from and the actual text. This runs directly contrary to Naives Bayes models that typically assume independence between all features. As demonstrated, the present invention assumes independencies between some features and dependencies between other features.

Referring now to Fig. 4, there is illustrated a flow diagram of an exemplary logic process 400 that can be employed when determining whether incoming messages are spam. The process 400 involves receiving a new message(s) at 410 and then extracting useful data from the message at 420. Such useful data can include IP address data and text-related data. Following therefrom, it can be determined at 430 whether the IP address is known – that is, has it been seen before in previous messages such that a particular filter would recognize it. If the IP address is known, then the IP address data can be run through a known IP address filter at 435.

Conversely, if the IP address is unknown at 430, then the method can determine if the top 24 bits of the IP address data are known at 440. This can be an important determination since it is possible that the IP address extracted from the message is incomplete. Alternatively, if an exact match to the IP address is unknown, but other messages having the same top 24 bits have been observed in previous messages, a filter trained on the top 24 bits of that IP address can be used to determine if the message is spam at 445. However, the top 24 bits may be unknown as well. In this case, it can be determined whether the top 16 bits of that IP address are known at 450. If yes, then the top 16 bits of the extracted IP address data can be run through a 16-bit trained filter at 455. If no, then the available IP address data can be run through a filter trained up on other messages received near the same time as the current message that also have unknown IP addresses at 460.

Concurrently or following therefrom, extracted text-based features can be run through a text-based feature filter at 470. The results (scores) from the filtering runs can be collected and then combined at 480. There are different ways to combine filter results from the at least two filter models. In one approach, the scores can be multiplied. Another approach calls for adding the scores together while a third approach assesses both scores based on the particular filter's effectiveness in detecting spam, for example. In other words, if an IP address model is about 90% effective and a text features model is about 10% effective, the overall effectiveness of the filtering scheme can be concluded to be about 80% effective because the IP address model can be trusted more, and hence given more value or weight in this determination. An additional filter can be trained to combine scores in such a manner. Finally, a message verdict can be obtained at 490.

Messages can include many features of a particular type, and in many cases, it is not clear which feature is the relevant one because spammers may try to confuse spam filters, thus rendering the features ambiguous. In particular, there are two cases. First, with URLs, a spammer may add as many URLs as he wants to the message. Second, there is the IP address. In many cases, such as on a server, the system can tell which is the last IP address external to the system, but this is relatively uncommon on a client. In particular, messages are typically sent from server to server to server. In some cases, especially email server software, the last IP address external to the recipient's organization can be clearly identified. Unfortunately, this is the only (external) IP address that the recipient can trust – since there is no reliable way to know which other IP addresses were actually used. On client software, the client will receive a list of IP addresses (in reverse order from which they were received.) He does not know which ones are external to his organization, and which ones are internal. As on the server, of the external IP addresses, he can only trust the last external address (highest external server on the list) – except he does not know which IP address this is.

As previously mentioned, the ambiguity problem also exists for URLs. Typically, a spammer will only send users to one URL, but he can include links to other domains, including known good domains. The links to the known good domains will typically be in places where users are unable to click on them, but where it is difficult for an automatic spam filter to tell the difference between the true link and the false ones.

There are a couple of common heuristics for finding the last external IP address. Unfortunately, none of them consistently work with reasonable accuracy. In one heuristic, the MX record for the recipient's organization is looked up. For example, if his email address is known to be name@yourdomain.com, the MX record, which contains the list of IP addresses for yourdomain's gateway servers for yourdomain.com can be looked up. The client then looks down through the received-from list until one of these addresses is found. Unfortunately, this does not always work because the inwards and outwards IP addresses may be different for the gateway box.

A similar heuristic uses internal IP addresses. Some IP address ranges are internal-only (those that start with 192, for instance). By going down through the list until one finds an IP address that is not definitely internal, perhaps one can find the first external one. Similarly, one can perform reverse IP lookups on each IP address in the list, and see if it maps to the same domain as the recipient. Eventually, one finds an IP address that fails both of these heuristics, and assumes it is the last external one. Unfortunately, these heuristics don't necessarily work either. For instance, some companies' servers use statically defined addresses, not the ones in the internal only ranges. Furthermore, the reverse-IP mapping is not always set up correctly. Finally, some domains owned by a common owner may use the same servers for both domains. Imagine that Zoomail and ZNN are domains which use the same serves for both domains and Zoobox is used to access messages on Zoomail or ZNN on a client. If Zoomail uses .zoomail.com for all its servers – then a Zoobox client for a ZNN user would find such a server on its list and assume it was external – so even requiring all admins to set up their internal server reverse IP addresses correctly would not work.

One algorithmic methodology that has been shown to work has been described in the related co-pending application (Serial No. 10/454,168). As stated therein, one way for spam filtering software to find this sender's IP address is to know the mail server configurations at an organization. In general, if one knows which machines pass to which other machines in which situations, one can determine the sender's IP address. However, especially for spam filtering software installed on email clients, it may not be convenient to describe the server configuration. An alternate approach involves utilizing MX records to determine the true source of a message. MX records list, for each domain,

the FQDNs (fully qualified domain names – the names of the machines) of recipients of email for that domain. These FQDNs can be mapped to their IP addresses, and we can trace back through the received from list until we find an IP address corresponding to an FQDN corresponding to an entry in the domain's MX record. The IP addresses that this machine received from is the sender's IP address. Imagine that 1.2.3.101 is the only MX record for x.com. Then by finding the line that received from 1.2.3.101, we know the next line corresponds to x.com's incoming mail server, and thus that the IP address in that line corresponds to the IP address that sent to x.com.

Line	*comment*
`Received: from a.x.com ([1.2.3.100]) by b.x.com Tue, 22 Apr 2003 13:11:48 -0700`	Internal to x.com
`Received: from mailserver.x.com ([1.2.3.101]) by b.x.com Tue, 22 Apr 2003 12:11:48 -0700`	1.2.3.101 is an MX record for x.com so we know next line is first internal to x.com
`Received: from outside.com ([4.5.6.7]) by mailserver.x.com Tue, 22 Apr 2003 11:11:48 -0700`	This is where x.com received the message; this is the last trusted line. Use 4.5.6.7 as sender's IP address
`Received: from trustedsender.com ([8.9.10.11]) by outside.com Tue, 22 Apr 2003 10:11:48 -0700`	This line may be fake, constructed by server at 4.5.6.7

Note that there is no currently accepted standard for listing outgoing mail servers, and note that this heuristic can fail if, for instance, IP addresses internal to an organization are different than those external to an organization, or if an organization sends mail from one machine listed in an MX record indirectly to another machine listed in an MX record. Note also that in the special case where the sender's IP as found above is found to be internal to the organization, as could happen if one machine in the MX record sent to another in the MX record, the process is continued as above. In addition, certain IP addresses can be detected as internal (because they are of the form 10.x.y.z or 172.16.y.z through 172.31.y.z or 192.168.0.z through 192.168.255.z, a form used only for

internal IP addresses); any address internal to an organization can be trusted. Finally, if a received from line is of the form "Received from a.x.com [1.2.3.100]" and an IP address lookup of a.x.com yields 1.2.3.100 or a reverse IP address lookup of 1.2.3.100 yields a.x.com and if x.com is the organization, then the next line can also be trusted.

Using these observations, it is often possible to find the sender's IP address. The exemplary pseudocode is as follows:

```
bool fFoundHostInMX;
if (external IP address of MX records matches internal IP address
of MX records)
{
    fFoundHostInMX = FALSE; # it's worth looking for
} else {
     # This is also the case to use when we don't know if the
internal and external addresses match.
     # If we use this case, we often return "don't know" but
that's better than being wrong
    fFoundHostInMX = TRUE; # it's not worth looking for, pretend
we already found it
}

for each received from line of the form Received from a.b.c
[i.j.k.l] {
    if i.j.k.l in MX records of receiver domain
    {
        fFoundHostInMX = TRUE;
        continue;
    }
    if not fFoundHostInMX
    {
        # Has not yet gone through an MX record, must be internal
        continue;
    }
    if i.j.k.l is of form
        10.x.y.z or
```

```
        172.16.y.z to 172.31.y.z or
        192.168.0.z to 192.168.255.z
    {
        # Must be internal
        continue;
    }
    if DNS lookup of a.b.c yields i.j.k.l and b.c is receiver
domain
    {
        # Must be internal
        continue;
    }
    Output sender's alleged FQDN a.b.c and sender's actual IP
address i.j.k.k
}
If we reach here, then Error: unable to identify sender's alleged
FQDN and sender's actual IP address
```

Unfortunately, much of the time, the previous algorithmic methodology simply concludes that it is unable to find the sender's actual IP address with confidence.

Turning now to Fig. 5, there is a flow diagram of one exemplary technique 500 that may work. The technique 500 involves a machine learning system attempting to learn which IP addresses are internal and which are external. For instance, on an email client, we can keep track of which messages a user marks as spam, and which messages a user marks as good at 510. When the user identifies a message as spam, it can be determined that at least one of the two or more IP addresses is bad at 520. At 530, we can then try to find an assignment of IP addresses (or hostnames from reverse IP lookups) to internal and external that maximizes the probability of the training data, or that maximizes accuracy on the training data. Following, at 540, the weight of the bad IP address can be adjusted for being an external IP address.

Referring now to Fig. 6, there is illustrated a flow diagram 600 of another option that can try to detect, using for instance a text-only model, good messages and spam at 610, and to use this as an input to the learning of which IP addresses are internal or external at 620. For instance, good messages do not include forgeries, and so may make

it clear, *e.g.*, from domain names, which are the external IP addresses, thus making it clear which are the recipient's edge servers.

Yet another option is to assume that the worst scoring IP address or worst scoring URL in a message is the most accurate predictor. That is, let SCORE(input) represent the score of an INPUT (*e.g.*, an IP address or URL). High values are considered bad (more spammy) in the present system. So, then the overall score returned for the IP part of a message in our system with *n* incoming IP addresses would be MAX(SCORE(IP_1), SCORE(IP_2), ..., SCORE(IP_n)). The score function might be a constant for each IP address. In other words, there is a list of IP addresses and their corresponding scores, or it might be a function in its own right, *e.g.*, we use information such as the IP address, the top 24 bits of the IP address, reverse DNS lookups on the IP address, etc., as features to a scoring function. The same can be done for URLs.

It is typically hard to do machine learning for functions that are not smoothly differentiable, such as the MAX function. However, the following describes a technique for doing such learning. This algorithm can be written as if it is for messages and IP addresses and features of IP addresses, but it will be evident that the same algorithm can be used for inputs other than email messages, *e.g.*, any "instance" for a machine learning system, and it will be evident that the properties of interest are not limited to IP addresses, but could be URLs, words, or any properties of the training instance that may occur more than one time within a training instance.

The algorithm that is proposed is as follows:

Let MAXSCORE(mk) be the overall score of message k.

Let IPk_1 IPk_2, ... IPk_{kl} represent the IP addresses in message k.

Then MAXSCORE(mk) = MAX(SCORE(IPk_1), SCORE(IPk_2), ..., SCORE(IPk_{kl})).

Assume there is an objective function that we are trying to maximize (*e.g.*, maximize accuracy on the training data, maximize the log probability of the training data, maximize a combination of accuracy and negative sum of square of weights, etc.). Call this function OBJECTIVE(m1...mk, w1...wn): it is a function of the weights and the training messages.

Assume that OBJECTIVE is a differentiable function of the weights.

Assume that the SCORE function is the sum of the weights. The exemplary pseudocode is as follows:

```
for each iteration until enough iterations or convergence {
    for each feature f_i {
        let m1...mp be the training instances (e.g., messages)
        that have feature f_i in one of their IP addresses {
            let MAXSCORE-fi(mq, w) be the value that
            MAXSCORE(mq) would have if feature i had weight
            w, and all other weights were unchanged.
            for each message mq, find the weight vq for f_i
            such that MAXSCORE-f_i(mq, w) shifts from being
            independent of w to being dependent on w (if all
            IP addresses in mq have feature f_i, then vq = -
            infinity.)  In particular, vq = limit as w
            approaches infinity of  MAXSCORE-f_i(mq, -
            infinity) - MAXSCORE-f_i(mq,w) + w.
            sort all messages mq by the weight vq in
            increasing order.  (Assume that mq and vq refer
            to the sorted order.)
            now, for the current feature f_i, for values of w
            between vq and vq+1, OBJECTIVE(m1...mn, w1...wq-
            1,w,wq+1,...wn) is differentiable; use standard
            means to solve for the best value.  Note that
            depending on the form of OBJECTIVE, even more
            efficient techniques may be possible.  For
            instance, if OBJECTIVE is simply the function
            that gives one point for each correctly
            classified message, then a binary search across
            all values can be used, and only values w = vq
            for some q will change the objective function
        }
    }
}
```

Fig. 7 illustrates the above algorithmic methodology 700 as a flow diagram. In particular, for each feature f_i, let m1 up to mp be the training instances or messages that have feature f_i in one of their IP addresses at 710. At 720, find the weight (vq) for f_i where the message would shift from being independent of f_i's weight to dependent on f_i's weight – for each message. At 730, sort the messages by weight (vq). At 740, for f_i, solve for the best value for values in a given range of weights. At 750, iterate over each IP address feature and learn how to adjust the weights of each feature to optimize the weight of any one IP address feature at any given time.

Thus, the method 700 is concerned with optimizing the weights associated with one feature at any given time. Messages that have that feature can be found; thus any other messages are not a concern. Essentially, the method wants to find the best weight for a feature. If it is a really high weight, it will be higher than everything else so as this weight changes, every other weight can change as well. However, if it is a really low weight, it will be lower than everything else and changing it will not change any other weight. So for each message, finding the value of that weight (vq) which the message would be dependent on is desired. Once the vq values for each message are determined, the messages can be sorted accordingly. Instead of checking different ranges and looking within each range to find the best value of vq, the change points can be looked at since these can be true points of interest.

Yet another technique for learning such functions is to notice that $MAX(a_1, a_2, \ldots, a_n)$ is approximately equal to $SUM(a_1^x, a_2^x, \ldots, a_n^x)^{(1/x)}$ where the approximation becomes more accurate the larger the value of x. One can use this function instead of the MAX function to get a function that is differentiable, and then one can use standard gradient descent style optimization techniques. Gradients may become steep for large values of x, so one may wish to initially use a small value of x, *e.g.,* 1 or 2, perform gradient descent, and then increase x and repeat.

When an IP address or URL is present and known, it can typically be one of the most valuable features in the message. Machine learning systems may not naturally assign more weight to these known IP addresses and URLs than to other features, such as known words. The system must be told in some way that these weights are more useful than other weights. This can be referred to as a value problem.

The value problem can be addressed through smoothing, which is also known as regularization. Typical machine learning systems smooth their parameters in some way. To give five examples, neural networks often use weight decay; maximum entropy models (also known as logistic regression) can use a Gaussian or Exponential prior; Support Vector Machines (SVMs) typically have a term minimizing the sum square of the weights, or the sum of the absolute values of the weights; and decision trees are often smoothed with a Dirichlet prior.

For all of these techniques, it is possible to modify the regularization so that the IP addresses or URLs are counted as more valuable. In addition, the smoothing can vary for sub-portions of features such as IP addresses or URLs. For example, different smoothing values can be employed for all 32-bits, a top 24-bits, and a top 16-bits of an IP address.

Fig. 8 provides a schematic diagram of the various smoothing techniques as modified by three machine learning systems to altering the associated smoothing effects. For instance, with a neural network 810, one can decrease the amount of weight decay on weights from IP addresses or URLs at 815. For Support Vector Machines 830, the formula usually optimized is (835):

$$\sum w_i^2 + c \sum penalty(score(message(j)), truth(j))$$

where penalty is a penalty function.

This can be rewritten as:

$$\frac{1}{c}\sum w_i^2 + \sum penalty(score(message(j)), truth(j)).$$

If we want, we can use different weights for different features. Instead of having one value of c, we can have one value of c_i for each weight w_i:

$$\sum \frac{1}{c_i} w_i^2 + \sum penalty(score(message(j)), truth(j)).$$

With maximum entropy models 820, one can use a larger standard deviation for the Gaussian prior at 825. For maximum entropy models, the formula usually optimized is $\left(\prod_i P(y_i \mid x_i; \Lambda)\right) \times \left(\prod_j P(\lambda_j)\right)$.

That is, the first term is the probability of the training data, given the parameters of the model (Λ). The second term is the probabilities of each parameter. The

probability of a parameter is defined by, for instance, a Gaussian distribution, typically with mean 0, and standard deviation σ_i. We can make the value of σ_i be different for different parameter types, including making it larger for IP address or URL features.

Typically, features are assigned values of 0 or 1. For instance, the presence of an IP feature such as 1.2.3.4 might result in the presence of a feature f_{1000} that has value 1 when the IP address is present and 0 when it is not. It will be recognized by those skilled in the art of machine learning that using a larger value for f_{1000}, e.g. $f_{1000} = 10$ when the IP address is present, 0 when it is not, will have a similar or identical effect in many model types, including SVMs and maximum entropy models, as changing the smoothing explicitly.

Note that in practice for IP addresses, features based not just on the IP address, but based also on the top 24 and top 16 bits, or other combinations of bits can be employed. Note that for URL features portions of the URL, such as the fully qualified domain name, and sub-domains of the fully qualified domain name can also be used. The systems and method described hereinabove can also be combined with quarantining techniques (*e.g.,* suspending delivery of messages until more information can be collected about the messages) and with a feedback loop, whereby participating users provide feedback in the form of classifications of messages to facilitate updating and/or building filters.

In order to provide additional context for various aspects of the present invention, Fig. 9 and the following discussion are intended to provide a brief, general description of a suitable operating environment 910 in which various aspects of the present invention may be implemented. While the invention is described in the general context of computer-executable instructions, such as program modules, executed by one or more computers or other devices, those skilled in the art will recognize that the invention can also be implemented in combination with other program modules and/or as a combination of hardware and software.

Generally, however, program modules include routines, programs, objects, components, data structures, *etc.* that perform particular tasks or implement particular data types. The operating environment 910 is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or

functionality of the invention. Other well known computer systems, environments, and/or configurations that may be suitable for use with the invention include but are not limited to, personal computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include the above systems or devices, and the like.

With reference to Fig. 9, an exemplary environment 910 for implementing various aspects of the invention includes a computer 912. The computer 912 includes a processing unit 914, a system memory 916, and a system bus 918. The system bus 918 couples the system components including, but not limited to, the system memory 916 to the processing unit 914. The processing unit 914 can be any of various available processors. Dual microprocessors and other multiprocessor architectures also can be employed as the processing unit 914.

The system bus 918 can be any of several types of bus structure(s) including the memory bus or memory controller, a peripheral bus or external bus, and/or a local bus using any variety of available bus architectures including, but not limited to, 11-bit bus, Industrial Standard Architecture (ISA), Micro-Channel Architecture (MSA), Extended ISA (EISA), Intelligent Drive Electronics (IDE), VESA Local Bus (VLB), Peripheral Component Interconnect (PCI), Universal Serial Bus (USB), Advanced Graphics Port (AGP), Personal Computer Memory Card International Association bus (PCMCIA), and Small Computer Systems Interface (SCSI).

The system memory 916 includes volatile memory 920 and nonvolatile memory 922. The basic input/output system (BIOS), containing the basic routines to transfer information between elements within the computer 912, such as during start-up, is stored in nonvolatile memory 922. By way of illustration, and not limitation, nonvolatile memory 922 can include read only memory (ROM), programmable ROM (PROM), electrically programmable ROM (EPROM), electrically erasable ROM (EEPROM), or flash memory. Volatile memory 920 includes random access memory (RAM), which acts as external cache memory. By way of illustration and not limitation, RAM is available in many forms such as synchronous RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR SDRAM), enhanced

SDRAM (ESDRAM), Synchlink DRAM (SLDRAM), and direct Rambus RAM (DRRAM).

Computer 912 also includes removable/nonremovable, volatile/nonvolatile computer storage media. Fig. 9 illustrates, for example a disk storage 924. Disk storage 924 includes, but is not limited to, devices like a magnetic disk drive, floppy disk drive, tape drive, Jaz drive, Zip drive, LS-100 drive, flash memory card, or memory stick. In addition, disk storage 924 can include storage media separately or in combination with other storage media including, but not limited to, an optical disk drive such as a compact disk ROM device (CD-ROM), CD recordable drive (CD-R Drive), CD rewritable drive (CD-RW Drive) or a digital versatile disk ROM drive (DVD-ROM). To facilitate connection of the disk storage devices 924 to the system bus 918, a removable or non-removable interface is typically used such as interface 926.

It is to be appreciated that Fig. 9 describes software that acts as an intermediary between users and the basic computer resources described in suitable operating environment 910. Such software includes an operating system 928. Operating system 928, which can be stored on disk storage 924, acts to control and allocate resources of the computer system 912. System applications 930 take advantage of the management of resources by operating system 928 through program modules 932 and program data 934 stored either in system memory 916 or on disk storage 924. It is to be appreciated that the present invention can be implemented with various operating systems or combinations of operating systems.

A user enters commands or information into the computer 912 through input device(s) 936. Input devices 936 include, but are not limited to, a pointing device such as a mouse, trackball, stylus, touch pad, keyboard, microphone, joystick, game pad, satellite dish, scanner, TV tuner card, digital camera, digital video camera, web camera, and the like. These and other input devices connect to the processing unit 914 through the system bus 918 *via* interface port(s) 938. Interface port(s) 938 include, for example, a serial port, a parallel port, a game port, and a universal serial bus (USB). Output device(s) 940 use some of the same type of ports as input device(s) 936. Thus, for example, a USB port may be used to provide input to computer 912 and to output information from computer 912 to an output device 940. Output adapter 942 is provided to illustrate that there are

some output devices 940 like monitors, speakers, and printers among other output devices 940 that require special adapters. The output adapters 942 include, by way of illustration and not limitation, video and sound cards that provide a means of connection between the output device 940 and the system bus 918. It should be noted that other devices and/or systems of devices provide both input and output capabilities such as remote computer(s) 944.

Computer 912 can operate in a networked environment using logical connections to one or more remote computers, such as remote computer(s) 944. The remote computer(s) 944 can be a personal computer, a server, a router, a network PC, a workstation, a microprocessor based appliance, a peer device or other common network node and the like, and typically includes many or all of the elements described relative to computer 912. For purposes of brevity, only a memory storage device 946 is illustrated with remote computer(s) 944. Remote computer(s) 944 is logically connected to computer 912 through a network interface 948 and then physically connected *via* communication connection 950. Network interface 948 encompasses communication networks such as local-area networks (LAN) and wide-area networks (WAN). LAN technologies include Fiber Distributed Data Interface (FDDI), Copper Distributed Data Interface (CDDI), Ethernet/IEEE 1102.3, Token Ring/IEEE 1102.5 and the like. WAN technologies include, but are not limited to, point-to-point links, circuit switching networks like Integrated Services Digital Networks (ISDN) and variations thereon, packet switching networks, and Digital Subscriber Lines (DSL).

Communication connection(s) 950 refers to the hardware/software employed to connect the network interface 948 to the bus 918. While communication connection 950 is shown for illustrative clarity inside computer 912, it can also be external to computer 912. The hardware/software necessary for connection to the network interface 948 includes, for exemplary purposes only, internal and external technologies such as, modems including regular telephone grade modems, cable modems and DSL modems, ISDN adapters, and Ethernet cards.

What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill

in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications, and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a transitional word in a claim.